Automatic Flower Detection and Classification System Using a Light-Weight Convolutional Neural Network

Johanna Ärje[a,b,c], Dimitrios Milioris[d], Dat Thanh Tran[b], Jane U. Jepsen[e], Jenni Raitoharju[b], Moncef Gabbouj[b], Alexandros Iosifidis[f], Toke Thomas Høye[a]

[a] Department of Bioscience, Aarhus University, Denmark, johanna.arje@tuni.fi

[b] Department of Computing Sciences, Tampere University, Finland

[c] Department of Mathematics and Statistics, University of Jyvaskyla, Finland

[d] Nokia Bell Labs & MIT, USA

[e] Norwegian Institute for Nature Research, Norway

[f] Department of Engineering, ECE, Aarhus University, Denmark

***Abstract*—Phenology describes the timing of life history events like flowering in plants and is a sensitive indicator of biological responses to climate change. However, recording plant phenology across space and time is a labour-intensive task. The creation of autonomous systems for in situ monitoring could greatly reduce expensive and time consuming manual human labour to both collect and analyze the data. One of the bottlenecks in creating such an autonomous system is computational complexity of the adopted Computer Vision methods. Deep Convolutional Neural Networks (CNNs) are state-of-the-art object detectors but can be very slow and computationally demanding. Light CNN topologies with only few layers which can achieve good performance are needed for lowering the processing power requirements. In this paper, we compare a light-weight CNN with two deeper CNNs on an object detection as well as image classification task on a dataset of Dryas flowers from Greenland.**

***Index Terms*—automated biomonitoring, classification, convolutional neural networks, object detection, plant phenology**

I. Introduction

Climate warming is causing changes in the timing and duration of flowering seasons [1]–[3]. This effect of global warming is most readily observed in the Arctic where the changes are happening most rapidly. Also, the lower number of plant species and the two-dimensional structure of the vegetation make it convenient to study these changes in arctic areas. To study the changes in flowering, biologists typically monitor plants in small permanently marked areas and manually count buds, flowers and wilted flowers repeatedly during the growing season to estimate the start and end of the flowering season. With more efficient methods of recording plant phenology, it could become possible to track individual flowers and even to link such phenological dynamics at the individual level to visitations by pollinators, and ultimately to relate the timing and number of visits by pollinators to the likelihood of seed set in individual flowers. Automating the monitoring process would greatly reduce the cost of collecting and analyzing flowering and pollination data and would open up new insights into species interactions. It could also enable monitoring sites that are difficult to reach.

The automated monitoring of arctic flowers and pollinators involves three different machine learning tasks: detecting flowers, identifying the flowering stage, and detecting pollinators visiting flowers. The state-of-the-art approaches for object detection and image classification use deep learning models (CNNs, see e.g. [5], [6]). These models can produce high accuracy but are very slow due to large amount of parameters raising the computational costs [7]. Even faster methods, e.g., YOLO [8], SPP-CNNs [9], use CNN once and are still affected by the number of parameters. Also, YOLO struggles to detect small objects in groups [8]. In addition to the large number of parameters in deep CNNs, the computational costs are even higher when these complex models are used to detect and classify objects in big, high-resolution images.

Tran et al. [10] studied the detection of arctic *Dryas integrifolia* flowers and pollinators visiting them using CNNs for the detection task, but were less concerned about the processing speed. For automation to truly benefit arctic biomonitoring, the proposed system needs to analyze large amounts of monitoring images fast. In this paper, we concentrate on flower detection and classification. We propose a light-weight CNN for both tasks and use a sliding window approach for detecting flower locations in test images. We compare the light-weight network with two deeper CNN architectures – including ResNet-50 – in flower detection and classification tasks. We compare the models based on mean average precision (mAP) scores and accuracy, but also on their training and prediction times.

II. Data

The data for this work was collected from three time-lapse cameras placed over different *Dryas integrifolia* flower beds in Narsarsuaq, Greenland over summer seasons 2017 and 2018. The images were recorded with Wingscapes TimelapseCam Pro cameras and stored as 6080 x 3420 resolution JPEG images. To ensure sufficient data on the flowering season, the flower beds were photographed once a minute, resulting in three time series of images, one from each flower bed. Two

time series were captured over the summer season 2017 and one over the summer season 2018.

A subset of each time series was annotated using VGG Image Annotator software [12]. The subsets consist of several images per day, a total 1448 frames all together. In the annotation process, an operator drew a rectangle around every flower found in a single image and labeled it belonging to one of four classes: bud, flower, wilted flower, or seed pod (Figure 1). One time series of images from 2017 and one from 2018 were used for training and validating the models. The other time series from 2017 was kept separate for testing.

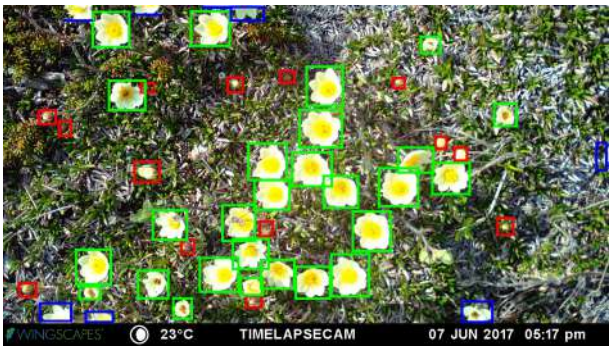


Fig. 1: Example of flower annotations. Red bounding boxes are drawn around buds, green boxes around flowers and blue boxes around any flower stage that is only partially visible in the image.

III. Flower detection and classification

We propose a two-step system for detecting and classifying *Dryas integrifolia* flower stages. In the first step, we train CNN models on a binary classification problem to separate any flower stage from the background. We train the networks with the annotation data, combining all flower categories into one, and background bounding boxes randomly sampled from the frames - making sure they do not overlap with the annotations. To detect the locations of flowers, we use the sliding window approach. We use several different window sizes to help detect objects of varying size and combine the resulting probability maps into one by taking the average over them. To obtain bounding box predictions, we first optimize a probability threshold that the predicted flower probabilities must surpass. Next, the threshold is used to set any uncertain pixels in probability map to zero and the remaining predicted probabilities to one. Finally, we place bounding boxes around all connected pixels to make our final flower location proposals and use non-maximum suppression to discard overlapping bounding boxes.

For flower stage classification, we use the same network architectures as for detection. In order for the networks to classify predicted bounding boxes well, we train and optimize them with predicted bounding boxes of the training and validation data instead of the original annotations. This time the training also uses all four flower categories (bud, flower, wilted flower and seed pod) and the background category. For each bounding box predicted by the detection network, we search the annotated bounding box with highest overlap and set the 'ground truth' label accordingly. The labels predicted by the flower classification networks are compared to these 'ground truth' labels.

A. Network architectures

1) The proposed light-weight model: For automatic flower monitoring with large amount of frames to go through, speed is a key factor in choosing the best network model. Deep CNNs can achieve high accuracy but are slow due to the high amount of parameters. Thus, we propose a Light-weight Convolutional Neural Network (LW-CNN) model [13]. It consists of three convolutional layers (32, 32 and 64 filters) and a fully connected layer (Fig. 2). The kernel size for all convolutional layers is 3×3. In order to learn the non-linear relations and extract the features, a fully connected layer of 128 neurons follows. The model uses 2×2 max-pooling with strides set to 2 in all dimensions. Biases are added to the results of the convolution layers, with a bias-value added to each filter-channel. Rectified Linear Unit (ReLU) has been used in order to add some non-linearity to the process, and it is executed after the pooling process. To predict the right class, we get the likelihood of the images which belong to each one of the classes and use the softmax function. We obtain the cost by calculating the cross entropy for each prediction.

Fig. 2: Description of the light-weight CNN

2) Models used for comparisons: We compare our LW-CNN network to two other network architectures. ResNet-50 is a deep CNN that uses residual learning [14]. It consists of 50 residual layers making it a very deep network with a large amount of parameters. For this study, we use a pre-trained ResNet-50 with a sigmoid activation function for the output layer. The pre-trained weights are finetuned with the training data.

The second network for comparison was used in [10] for a different subset of the data presented in this work. It follows the architechture of [15] with 12 convolutional layers and a global average pooling layer, and softmax activation function at the output of the network. The model uses dropout of 0.2 after convolutional layers 3, 6 and 9.

IV. Experimental results

A. Experimental setup

For the flower detection task, we combined all four flower stage categories into one. To train the models, we randomly sampled patches of the image background, making sure they do not overlap with the annotated bounding boxes by more than 20 %. The background patches were set to have width and height of 70 % of the maximum width (729 px) and height (713 px) of the annotated bounding boxes. To maintain the

scale, if an annotated bounding box was found to be smaller, it was extended to these dimensions as well.

As the images are of high resolution, we scaled them to 0.01 size to save time training and testing the model. The annotated bounding boxes and background patches were scaled to size 64×64 px before feeding them to the CNN models.

The aim of this automated flower monitoring system is to track individual flowers. As flowers on the edge of the frames are sometimes visible and sometimes not, we created a buffer on the edges of the image. The buffer was set to half of the maximum dimensions of the annotated bounding boxes in the training data, i.e., half of the maximum width at the sides and half of the maximum height at the top and bottom. Any box with a center outside the buffer was left out of the training data.

To enhance the performance of the detection networks, we used data augmentation on the training patches. Before each epoch, all training patches were randomly moved around a maximum 10 % in each direction, randomly flipped horizontally, and randomly zoomed in to a maximum scale of 70 %.

We used two flower image time series for training and validation. Approximately 70 % of the frames in these time series were selected for training and the rest for validation. The training images were selected randomly while trying to keep the proportion of different categories as even as possible. The third time series was used for testing. Table I shows the label distribution for training, validation and test data.

TABLE I: Label distribution

Data partition	Bud	Flower	Wilted flower	Seed pod	Frames
Training	886	1650	1782	1057	879
Validation	770	2482	2748	481	416
Testing	410	1026	1213	2057	153

Once the models had been trained, we used them to detect the locations of flowers in the test images. The test images were scaled to 0.01 size and a sliding window approach with a stride of 5 px was used to go over the image and predict flower probabilities. To detect objects of different scales, each test frame was processed with four sliding window sizes (45, 55, 65, 75). The predicted probability maps were averaged and the average map was upsampled back into the original image size of 6080×3420 px. We optimized the probability threshold that the predicted flower probabilities must surpass by maximizing the average Intersection over Union (IoU) for annotated and predicted bounding boxes in the validation frames. As with annotated boxes, if the center of the predicted box fell outside the buffer the box was disregarded.

For flower stage classification, the networks were trained and optimized with predicted bounding boxes of the training and validation data. For each predicted bounding box, we searched the annotated bounding box with the highest overlap and set the 'ground truth' label accordingly. If no annotated bounding box overlapped with the predicted box with Intersection over Minimum (IoM) $\geq$ 0.3, we set the 'ground truth' label to background. As the 5-category classification task is a more complex one, we increased the augmentation of the training data by randomly moving the bounding boxes a maximum 40 % in each direction before each epoch. The horizontal flipping and random zooming in remained the same as before.

The detection networks are compared based on training and prediction time, memory required for training, mAP scores, the number of False Positives Per Image (FPPI) and the average percentage of misdetected objects, i.e., False Negative (FN). The flower classification networks are compared based on classification accuracy.

All three networks were built using Keras [16]. The number of epochs and the learning rate for all models were optimized based on validation accuracy. We trained the detection networks for 50 epochs for each learning rate in (0.001, 0.0001,0.00001, 0.000001), 200 epochs all together. The classification networks were trained for 200 epochs for each learning rate. All the layers of the pre-trained ResNet-50 were finetuned for the same number of epochs. Both training and testing were run on a NVIDIA Tesla K80 GPU.

B. Flower detection results

Table II shows the accuracy, detection statistics and computation times for the flower detection networks. All three CNNs give similar results in terms of binary classification. The validation and test accuracy in Table II are calculated for known locations, i.e. patches of either background or annotated patches with a flower object in the center, and the accuracy is close to 100 % for all three networks. However, keeping the number of epochs fixed, increasing the depth and complexity of a network substansially increases the computing time for training the network. Training a ResNet-50 architecture with 50 layers takes approximately four times as long as training the LW-CNN with only three convolutional layers - both achieving similar classification accury.

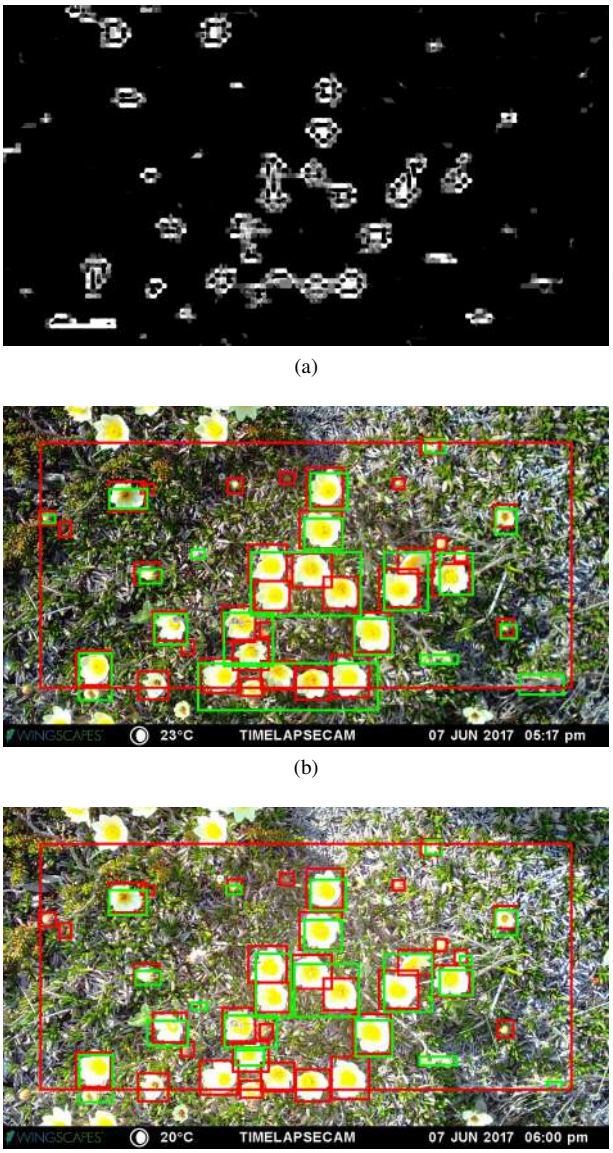
With the dataset being reasonably small, the training time of any of the three networks is not a challenge. However, considering that the aim is to use a chosen network to go through hundreds of thousands of time-lapse camera frames, the time used for testing is much more critical. Using the sliding window approach to produce a probability map of flowers for one test frame slows down considerably with the increasing complexity of the network.

While all the networks produce excellent classification results, the detection of flower locations is not as easy. As we draw bounding box predictions around connected areas in the probability map, flowers growing close to each other are easily enclosed in one large bounding box covering multiple flowers. This issue can be seen in Fig. 3a where human eye can easily distinguish the five flowers at the bottom of the frame but as they are connected they will all be enclosed in one large bounding box (Fig. 3b).

The problem of separating the flowers is also reflected in the mAP scores of Table II. Our approach works best with the LW-CNN as it has the highest mAP score due to it being the best of the three networks at separating close-by flowers, but it

TABLE II: Performance statistics for flower detection networks. Accuracy is given for annotated validation (ACC_{val}) and test (ACC_{test}) bounding boxes. Detection statistics are mAP, average number of FPPI, and % of FN predictions. Total number of annotated objects in the 153 test frames is 4713.

Model	Training time (on GPU)	Req. memory	Known locations ACC_{val}	Known locations ACC_{test}	Testing time per image (on GPU)	mAP	FPPI	FN %
ResNet-50 [14]	04:18:05	3.4 GB	99.45 %	95.76 %	0:10:22	0.35	6.91	7.25 %
CNN [15]	03:34:47	2.9 GB	99.41 %	97.29 %	0:02:43	0.36	0.90	13.98 %
LW-CNN [13]	01:11:31	2.7 GB	99.39 %	96.46 %	0:01:22	0.52	3.20	13.87 %



(a)

(b)

(c)

Fig. 3: A mask and two test frame predictions of the LW-CNN. The binary mask is formed by setting all probabilities above a set threshold to one. Annotated bounding boxes are shown in red and predicted bounding boxes in green. The big red rectangle represents a buffer where any annotated or predicted bounding box with its center outside the buffer will be left out. This behaviour is demonstrated as in (a) the large green box at the bottom is included while in (b) it has been left out.

does produce on average 3.2 false positive objects per frame. The mAP is a good measure of performance in retrieval tasks where all objects are assumed to be retrieved. However, this is not the case with our detection task as can been seen by the percentage of false negative predictions. The percentage of objects not detected is higher than in [10] because of the use of the buffer at the frame edges. While leaving out all training objects with their center outside the buffer, we also leave out any predicted bounding boxes with the same criteria. Due to the issue of our methods not separating close-by flowers, many predicted bounding boxes containing flowers close to the buffer tend to have their center outside the buffer and are therefore left out of the final predictions. This phenomenom can be seen in Fig. 3b, 3c with the row of flowers near the bottom buffer.

C. Flower classification results

The classification results for flower stages are presented in Table III. Again, classification accuracies for annotated boxes of the validation data are very good. Classifying flower stages for the test data proves a more difficult task with accuracies decreasing substansially for predicted flower locations.

Table IV shows confusion matrices for classifying the flower stages of predicted flower locations. The open flowers are identified well while wilted flowers and seed pods are easily mixed. The bud stage is the most difficult category to classify but interestingly they are not confused with the background but with seed pods. The LW-CNN does a very good job identifying the background boxes considering that those were false positive predictions of the detection network.

V. Conclusions

In this work, we propose an automated monitoring system for arctic Dryas flowers with time-lapse images. Our proposed system works in two steps, a binary network for detection and a second network for classifying the detected objects. We explore a light-weight CNN for detecting and classifying the flower stages and compare the network structure with two deeper CNN architectures in terms of performance and computation time. The networks produce similar classification accuracies but have differences in detecting flowers. The light-weight network yields the highest mAP score for detection, and the highest classification accuracy for predicted flower locations. While the training time of the networks is quite fast, the time used for detecting flowers in the test frames is ten-fold for the deep ResNet-50 architecture compared to the proposed light-weight network. As automated flower monitoring would

TABLE III: Performance statistics for flower stage classification networks. Accuracy is given for predicted validation (ACC_{val}) and test (ACC_{test}) bounding boxes. Total number of annotated objects in the 153 test frames is 4713 while the number of predicted objects varies depending on the network.

Model	Training time (on GPU)	Predicted locations ACC_{val}	Predicted locations ACC_{test}
ResNet-50 [14]	09:24:43	90.60 %	58.32 %
CNN [15]	04:06:16	78.86 %	65.54 %
LW-CNN [13]	02:05:09	90.27 %	69.15 %

TABLE IV: Confusion matrices for flower stage classification of predicted flower locations in test frames. (B = Bud, F = Flower, W = Wilted flower, S = Seed pod, X = Background)

		Predicted categories														
		ResNet50 [14]					CNN [15]					LW-CNN [13]				
True categories		B	F	W	S	X	B	F	W	S	X	B	F	W	S	X
	B	39	5	78	2	63	0	2	13	4	0	30	5	102	6	30
	F	3	269	75	0	12	0	387	96	29	15	2	478	82	7	13
	W	0	10	267	6	114	4	31	197	149	18	1	25	395	96	46
	S	5	11	556	120	39	28	23	323	1128	36	3	7	333	999	53
	X	15	55	38	2	829	1	20	32	83	13	13	64	74	30	322

likely involve processing tens of thousands, or even millions, of images, this renders the more complex networks unsuitable. Although the testing time with the sliding window approach is quite long even for the light-weight network, parallelizing the computation on multiple GPU nodes would speed up the process considerably.

The most difficult part of the detection and classification process in our approach is separating overlapping and close-by flowers as even a single connecting pixel results in large predicted bounding boxes covering multiple objects. The connected flower predictions could be due to the low resolution we use in orded to speed up the detection process. While most flowers in the frame are detected, tracking individual flowers is cumbersome if all the flowers are not identified separately. This also complicates the flower stage classification as a predicted bounding box can cover objects of different stages but will only receive one stage prediction. Hence, we found it better to also train the classification networks with predicted bounding boxes.

The classification of predicted flower locations proved challenging. However, some classification errors may not be a problem because of the way the data is collected as time-lapse images. With knowledge of the sequence of life history events, we could post process the predictions to disable going backwards in flowering stages. Also, as flowering is a gradual process it can be difficult even for experts to distinguish between the stages of open flowers and wilted flowers. For future work, we could try combining these two stages. In addition, the challenging bud stage could be omited.

While there remain issues to be solved before the automated flower monitoring system can be introduced in practice, automating the monitoring process is a necessary step. The rapid rate of changes due to global warming requires more efficient biomonitoring. In addition, an automated system enables researchers to monitor details that cannot be monitored manually, e.g., tracking individual flowers through the flowering stages and pollinator visitation.

ACKNOWLEDGMENT

The authors would like to thank the Villum Foundation (grant no. 17523) for funding this work and CSC Finland for computational resources.

REFERENCES

[1] Høye, T. T., Post, E., Schmidt, N. M., Trøjelgaard, K., and Forchhamm, M. C. (2013). Shorter flowering seasons and declining abundance of flower visitors in a warmer arctic. *Nature Climate Change*, 3:759–763.

[2] Prevéy, J. et al. (2017). Greater temperature sensitivity of plant phenology at colder sites: implication for convergence across northern latitudes. *Global Change Biology*, 23:2660–2671.

[3] Prevéy, J. et al. (2019). Warming shortens flowering seasons of tundra plant communities. *Nature Ecology and Evolution*, 3:45–52.

[4] Ärje, J. et al. (2017). The effect of automated taxa identification errors on biological indices. *Expert Systems with Applications*, 72:108–120.

[5] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 91–99.

[6] Liu, Y., Zhou, D., Tang, F., Meng, Y., and Dong, W. (2016). Flower classification via convolutional neural network. *IEEE International conference on Functional-Structural Plant Growth Modeling, Simulation, Visualization and Applications*, 110–116.

[7] Szegedy, C. et al. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

[8] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: unified, real-time object detection. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

[9] He, K., Zhang, X., Ren, S. and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:1904–1916.

[10] Tran, D. T., Høye, T. T., Gabbouj, M., and Iosifidis, A. (2018). Automatic flower and visitor detection system. *26th European Signal Processing Conference (EUSIPCO)*, 405–409.

[11] Waris, M. A., Iosifidis, A., and Gabbouj, M. (2017). Cnn-based edge filtering for object proposals. *Neurocomputing*, 266:631–640.

[12] Dutta, A., Gupta, A., and Zissermann, A. (2016). VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/.

[13] Milioris, D. (2019). Efficient Indoor Localization via Reinforcement Learning. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8350–8354.

[14] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[15] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv: 1412.6806*.

[16] Chollet, F. et al. (2015). Keras. https://keras.io.